Sincisco

ACM 答案

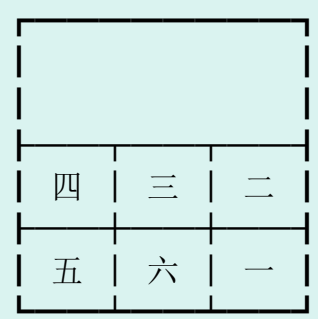
2010/11/10

Sincisco 小组成员

排球队员站位问题

i

【题目】排球队员站位问题



图为排球场的平面图，其中一、二、三、四、五、六为位置编号，二、三、四号位置为前排，一、六、五号位为后排。某队比赛时，一、四号位放主攻手，二、五号位放二传手，三、六号位放副攻手。队员所穿球衣分别为 1 ， 2 ， 3 ， 4 ， 5 ， 6 号，但每个队员的球衣都与他们的站位号不同。已知 1 号、 6 号队员不在后排， 2 号、 3 号队员不是二传手， 3 号、 4 号队员不在同一排， 5 号、 6 号队员不是副攻手。

编程求每个队员的站位情况。

【算法分析】本题可用一般的穷举法得出答案。也可用回溯法。

```
#include <cstdlib>
#include <iostream>
using namespace std;
int counter=0;
void display(int place[])
{
     for(int i=1;i<=6;i++)
     {
             cout<<place[i]<<"\t";
             if(i%3==0)
             cout<<"\n";
     }
     cout<<endl;
}
bool judge(int mark,int place[])
{
     switch(mark)
     {
                 case 1:
                      return true;
                 case 2:
                      return place[2]!=5&&place[2]!=6;
                 case 3:
                      return place[3]!=2&&place[3]!=3;
                 case 4:
                      return place[4]!=1&&place[4]!=2&&place[4]!=3&&place[4]!=6;
                 case 5:
                      return place[5]!=1&&place[5]!=5&&place[5]!=6;
                 case 6:
```

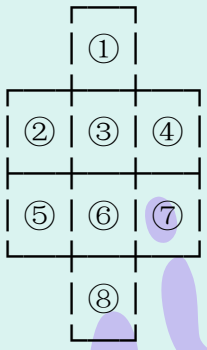
```
                    return place[6]!=1&&place[6]!=6;
    }
}
bool others(int place[])
{
    int m=1,n=1;
    for(int i=0;i<=6;i++)
    {
            if(place[i]==3)
            m=i-1;
            if(place[i]==4)
            n=i-1;
    }
    return m/3!=n/3;
}
void backtrace(int mark,int place[],int people[])
{
    for(int i=1;i<=6;i++)
    {
    if(people[i]!=0)
    {
    place[mark]=people[i];

    if(judge(mark,place))
    {
    if(mark==6&&others(place))
    {
    counter++;
    display(place);
    }
    else
    {
        people[i]=0;
        backtrace(mark+1,place,people);
        people[i]=i;
    }
    }
    }
    }
}
int main(int argc, char *argv[])
{
    int *place=new int[7];
    int *people=new int[7];
```

```
    for(int i=1;i<=6;i++)
    people[i]=i;
    backtrace(1,place,people);
    cout<<"The total methods is "<<counter<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

数字布置问题

【题目】 把 1-8 这 8 个数放入下图 8 个格中,要求相邻的格(横,竖,对角线)上填的数不连续.




```
#include <cstdlib>
#include <iostream>

using namespace std;
void display(int field[],int mark)
{
for(int i=1;i<=mark;i++)
cout<<field[i]<<"\t";
cout<<endl;
}
bool continues(int m,int n)
{
        if((m-n==-1)||(m-n==1))
        {
        return true;
        }
        else
        {
        return false;
        }
}
bool judge(int mark,int field[])
{
```

```
        switch(mark)
        {
            case 1:
                return true;
            case 2:
                return true;
            case 3:
                if(continues(field[3],field[1])||continues(field[3],field[2]))
                return false;
                else
                return true;
            case 4:
                if(continues(field[3],field[4]))
                return false;
                else
                return true;
            case 5:
                if(continues(field[5],field[2]))
                return false;
                else
                return true;
            case 6:
                if(continues(field[3],field[6])||continues(field[5],field[6]))
                return false;
                else
                return true;
            case 7:
                if(continues(field[7],field[4])||continues(field[7],field[6]))
                return false;
                else
                return true;
            case 8:
                if(continues(field[8],field[6]))
                return false;
                else
                return true;
        }
        }
void backtrace(int num[],int field[],int mark)
{
for(int i=1;i<=8;i++)
{
if(num[i]!=0)
{
```

```
                    field[mark]=num[i];
  if(judge(mark,field))
  {
  if(mark==8)
  display(field,mark);
  else
  {
  num[i]=0;
        backtrace(num,field,mark+1);
        num[i]=i;
  }
  }
  }
  }
  }
  int main(int argc, char *argv[])
  {
        int *num;
        int *field;
        num=new int[9];
        field=new int[9];
        for(int i=0;i<9;i++)
        {
        num[i]=i;
        }
        backtrace(num,field,1);
        system("PAUSE");
        return EXIT_SUCCESS;
  }
```


改错题

下列程序的功能是：一只甲虫在一个迷宫(一个 10×10 的矩阵表示)中移动，迷宫中有若干个柱子（在矩阵中表示为值为 1 的单元）。甲虫从迷宫中的某个位置出发，按照预先设置的指令在迷宫中行进：指令是一串由 1~4 组成的数字，分别代表行进的 4 个方向：1 代表向上，2 代表向右，3 代表向下，4 代表向左。当按照移动指令移动后的位置是一个柱子或超出了迷宫的范围则忽略该步指令。最终程序将显示指令序列结束后的甲虫所处的位置。

修改后的结果：

```
#include <iostream.h>

int m[5][5]={1, 0, 0, 1, 0,
                0, 1, 0, 1, 1,
           0, 0, 0, 0, 0,
```

```
          1,0,1,0,0,
          0,1,1,0,0};   //迷宫，1一柱子
int isPole(int x,int y)
{
  if(m[x][y]==1)
     return 1;
  else
     return 0;
}
void move(int d[],int x[2],int n)
{
  int nx, ny;
  for(int i=0; i<n ; i++)
  {
     switch(d[i])
     {
     case 1:
        ny = x[1]-1;
        break;
     case 2:
        nx = x[0]+1;
        break;
     case 3:
        ny = x[1]+1;
        break;
     case 4:
        nx = x[0]-1;
        break;
        default:
                  break;
     }
     if(isPole(nx,ny)||nx<0||nx>4||ny<0||ny>4)
        continue;
     x[0]=nx; x[1]=ny;
  }
}
void main()
{
  int x[2];
  int d[]={2, 2, 3, 3, 1, 3, 3, 4, 4, 1, 2, 3, 2, 2, 4};  //指令集
  cout<<"输入初始位置: ";
  cin>>x[0]>>x[1];
  move(d,x,sizeof(d[0])/sizeof(d));
  cout<<"The final position is: "<<x[0]<<", "<<x[1]<<endl;
```

}

2、给定一个整数 n，求出所有连续的且和为 n 正整数。比如对于整数 27，结果为 2~7、8~10、13 和 14，因为这些数之间的整数的和都是 27。注意：并不是所有的整数都有结果，例如不存在连续的整数和为 16。为了提高计算的效率，程序所采用的算法如下：(1) 从 1 开始计算连续的整数和 sum，直到 sum 不小于 n 为止；(2) 在第 i 步，如果 sum=i+(i+1)+...+j 比 n 大，则去掉连加的最左端的数 i，如果 sum 比 n 小，在连加的右端加上一个数(j+1)；(3) 如果和 sum=i+(i+1)+...+j 等于 n，则 i+(i+1)+...+j 为一组解，输出该解，并将连加的右端加上(j+1)；(4) 重复 2，3 步，直到 i 大于 n/2 为止。

```
#include <cstdlib>
#include <iostream>

using namespace std;
void dispaly(int left,int right,int num)
{
	cout<<"第"<<num<<"组解为：";
	for(int i=left;i<=right;i++)
	{
		cout<<i<<"\t";
	}
	cout<<endl;
}
void temp(int n)
{
	int left=1;
	int right=1;
	int sum=1;
	int num=0;
	for(;left<=n/2;)
	{
		if(sum<n)
		{
		right++;
		sum=sum+right;
		}
		if(sum>n)
		{
		sum=sum-left;
		left++;
		}
		if(sum==n)
		{
		num++;
```

```
                    dispaly(left,right,num);
                    sum=sum-left;
                    left++;
                    }

        }

}
int main(int argc, char *argv[])
{
    int n;
    cout<<"给定的整数为："<<endl;
    cin>>n;
    temp(n);
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

3、北大 3051

Satellite Photographs

Description

Farmer John purchased satellite photos of W x H pixels of his farm (1 <= W <= 80, 1 <= H <= 1000) and wishes to determine the largest 'contiguous' (connected) pasture. Pastures are contiguous when any pair of pixels in a pasture can be connected by traversing adjacent vertical or horizontal pixels that are part of the pasture. (It is easy to create pastures with very strange shapes, even circles that surround other circles.)

Each photo has been digitally enhanced to show pasture area as an asterisk ('*') and non-pasture area as a period ('.'). Here is a 10 x 5 sample satellite photo:

```
. . *. . . . . **
. **. . *****
. *. . . *. . . .
. . ****. ***
. . ****. ***
```

This photo shows three contiguous pastures of 4, 16, and 6 pixels. Help FJ find the largest contiguous pasture in each of his satellite photos.

Input

* Line 1: Two space-separated integers: W and H

* Lines 2..H+1: Each line contains W "*" or "." characters representing one raster line of a satellite photograph.

Output

* Line 1: The size of the largest contiguous field in the satellite photo.

Sample Input

```
10 5
..*.....**
.**..*****
.*...*....
..****.***
..****.***
```

Sample Output

```
16
```



```
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;
int largest=0;
int width;
int height;
bool judge(int m,int n,int **field)
{
        if(m>=0&&m<width&&n>=0&&n<height)
        {
        if(field[m][n]==1)
        return true;
        }
        return false;
}
void deep(int m,int n,int **field)
{
        field[m][n]=2;
```

```
        if(judge(m-1,n,field))
        deep(m-1,n,field);
        if(judge(m,n-1,field))
        deep(m,n-1,field);
        if(judge(m,n+1,field))
        deep(m,n+1,field);
        if(judge(m+1,n,field))
        deep(m+1,n,field);
}
void change(int **field)
{
        int m=0;
        for(int i=0;i<width;i++)
        for(int j=0;j<height;j++)
        {
                if(field[i][j]==2)
                {
                                        m++;
                                        field[i][j]=0;
                }
        }
        if(largest<m)
        largest=m;
}
int main(int argc, char *argv[])
{
      cin>>height>>width;
      int **field;
      field=new int*[width];
      for(int i=0;i<width;i++)
      {
      field[i]=new int[height];
      }
      string str;
            for(int i=0;i<width;i++)
      {
      cin>>str;
      for(int j=0;j<height;j++)
      {
      if(str[j]=='*')
      field[i][j]=1;
      else
      field[i][j]=0;
      }
```

```
        }
        for(int i=0;i<width;i++)
        for(int j=0;j<height;j++)
        {
        if(field[i][j]==1)
        {
                deep(i,j,field);
                change(field);
        }
        }
        cout<<"The size of the largest contiguous is "<<largest;
        system("PAUSE");
        return EXIT_SUCCESS;
    }
```

---

[i] 未经小组成员同意，不得随意在非 Sincisco 的其他群内发布